*HARSH KASHYAP  
CSE 4*

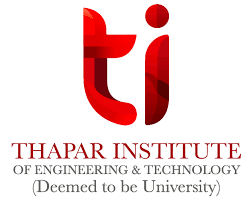
*101917088*

[*hkashyap\_be19@thapar.edu*](mailto:hkashyap_be19@thapar.edu)

A Practical activity Report submitted

for Operating Systems (UCS303)

**OPERATING SYSTEM**

****

Computer Science and Engineering

Patiala Campus

**2020**

Submitted to

Dr. Ashwani Kumar

**Assignment 4**

**Question 1**

**Basic file attributes : ls -l, -d option.**

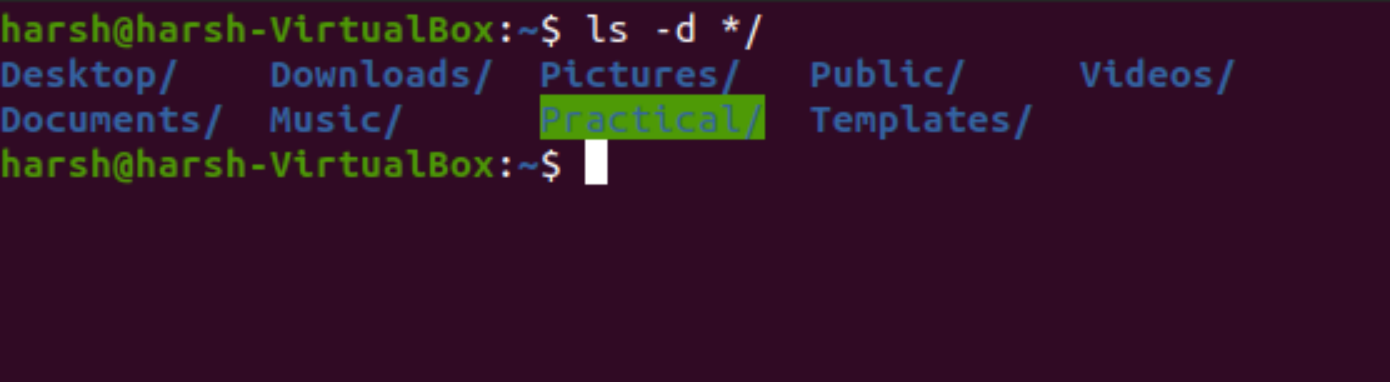
**Answer -**

Using ls -l character will display a long listing of the content of current directory ( i.e. not only prints the name of the file, but also some attributes such as owner, group owner, link count, permissions).

The first character shows the file type. The next nine characters are showing the file permissions. The first three characters are for the user, the next three are for the group, and the last three are for others.



Using ls -d character will display a long listing of the content of the directories present without showing the contents of the directory.This command is used to list only directories in the current directory. We have to type “ls -d \*/” command in the command window



**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 2**

**File Permission and Changing the access right.**

**Answer -**

In Linux operating systems, chmod is the command and system call which is used to change the access permissions of file system objects.it is used in other machines like servers, mainframe computers, supercomputers, etc.

Every file and directory in your Linux system has following 3 permissions defined for all the owners-

1. Read: This permission gives us the authority to open and read a file. Read permission on a directory gives you the ability to lists its content.
2. Write: The write permission gives us the authority to modify the contents of a file. The write permission on a directory gives you the authority to add, remove and rename files stored in the directory.
3. Execute: In Linux, we cannot run a program unless the execute permission is set.

Changing access rights - Using the “chmod” command, we can set permissions (read, write, execute) on a file/directory for the owner, group and the world. Syntax - chmod [reference][operator][mode] file.

The user can

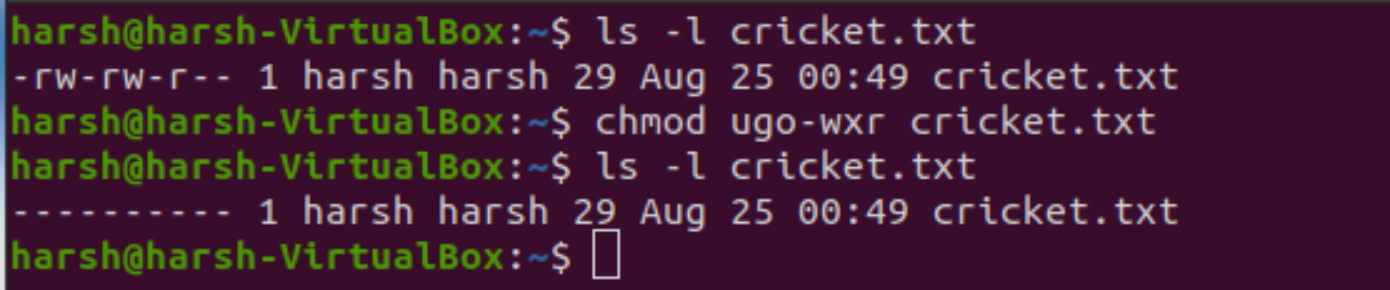
* read, write, and execute it;
* members of your group can read and execute it,
* others may only read it.

chmod u=rwx,g=rx,o=r myfile , letters u, g, and o stand for "user", "group", and "other".

We can also assign commands to files using numbers, each digit being a combination of the numbers 4, 2, 1, and 0:

* 4 stands for "read",
* 2 stands for "write",
* 1 stands for "execute",
* 0 stands for "no permission."

Removing all permission from cricket.txt which has read, write permission for user, read and write for a group and read for others.

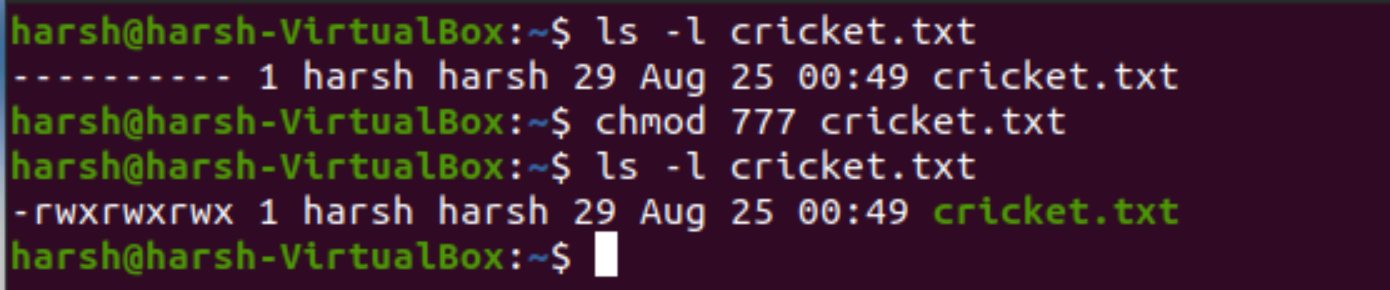


**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

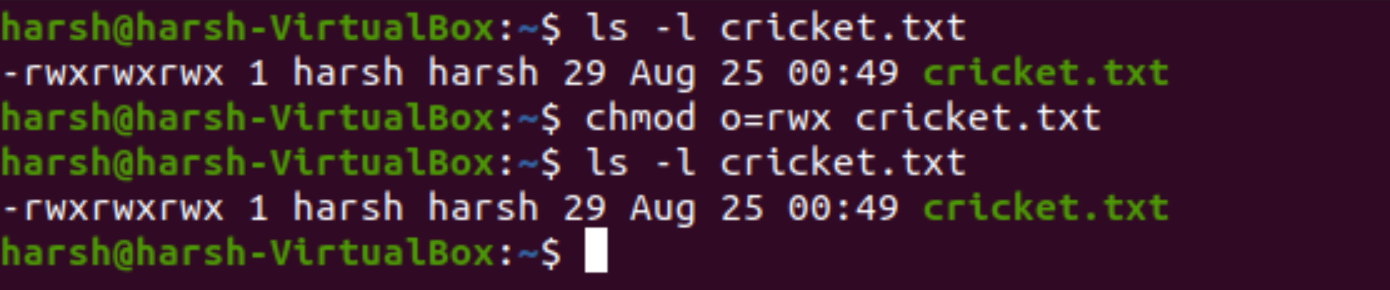
**Question 3**

**Relative and absolute permission**

**Answer -**

Absolute mode - In this mode, file permissions are not represented as characters but a three-digit octal number. So we use numbers to represent file permissions. This is the method which is most commonly used to set permissions.

Relative or Symbolic mode - In the symbolic mode, we can modify permissions of a specific owner. It makes use of mathematical symbols to modify the file permissions.



**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 4**

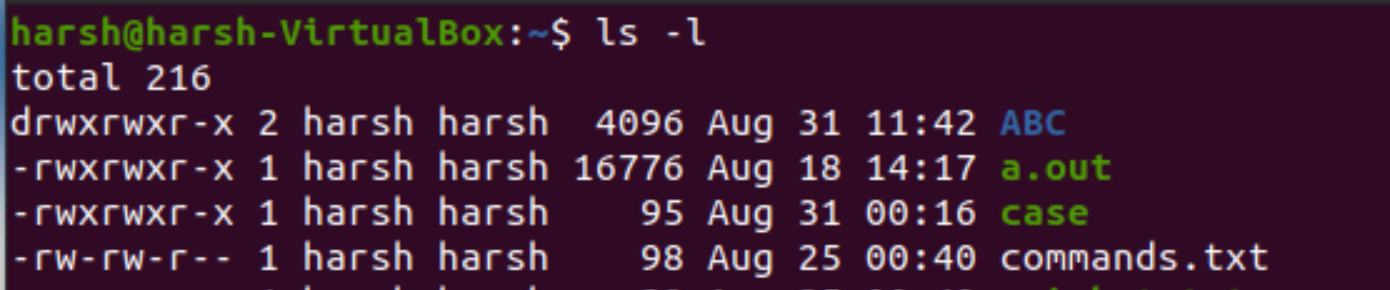
**Directory permission.**

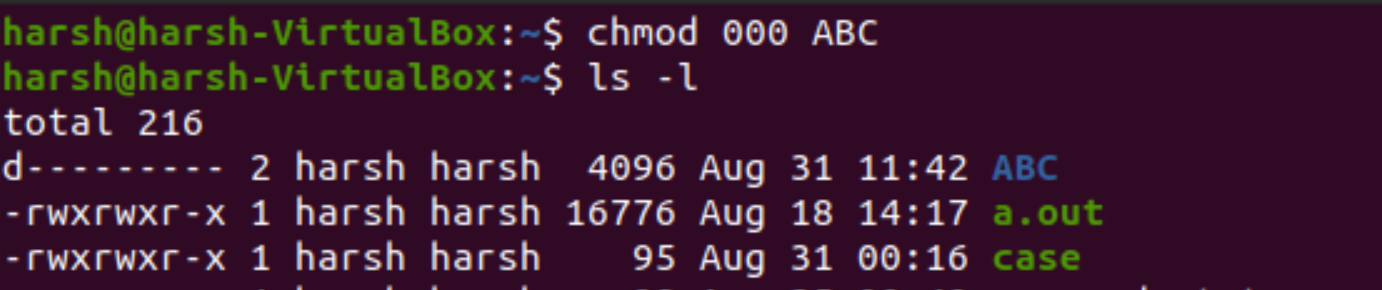
**Answer -**

Linux permissions dictate 3 things you may do with a file, read, write and execute. They are referred to in Linux by a single letter each.

* r read - you may view the contents of the file.
* w write - you may change the contents of the file.
* x execute - you may execute or run the file if it is a program or script.

For every file or directory, we define 3 sets of people for whom we may specify permissions. owner - a single person who owns the file. (typically the person who created the file but ownership may be granted to someone else by certain users) group - every file belongs to a single group. others - everyone else who is not in the group or is not the owner. Three permissions and three groups of people. That's about all there is to permissions really. Now let's see how we can view and change them.





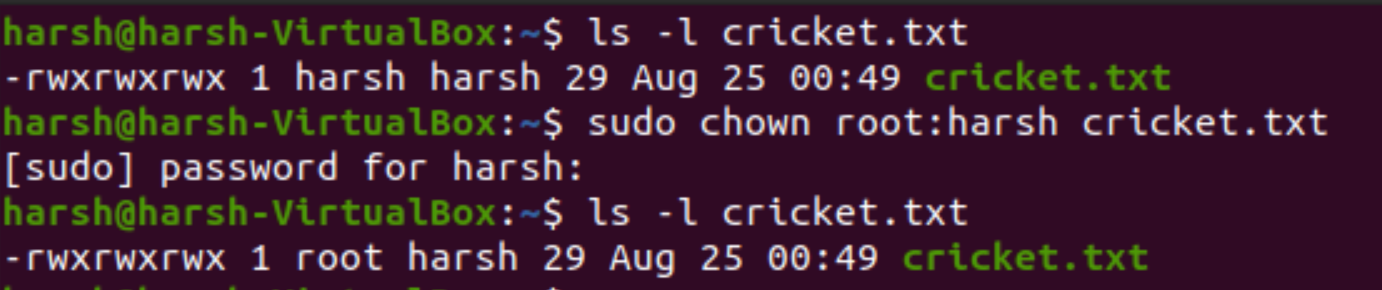
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 5**

**Changing file ownership.**

**Answer -**

Every file is owned by a specific user (or UID) and a specific group (or GID). The chown command can be used to change just the user or the user and group of a file. Here is an example of changing the owner of the file test to the user and its group to the user.



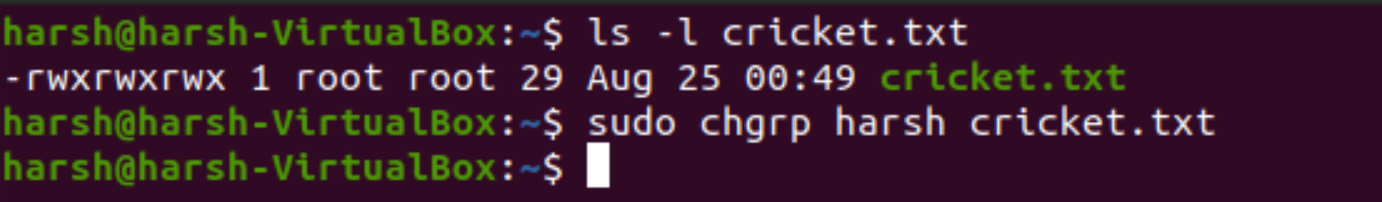
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 6**

**Changing file group.**

**Answer -**

In Linux, each file is associated with an owner and a group and has permissions that determine which users may read, write, or execute the file. Unlike the chown command that allows you to change the user and group ownership, chgrp changes only the group ownership. To find out to which group the file belongs to, use the ls -l command. Regular users can change the group of the file only if they own the file and only to a group of which they are a member. Administrative users can change the group ownership of all files.



**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Assignment 5**

**Question 1**

**vi editor and its basics.**

**Answer -**

vi is a screen-oriented text editor originally created for the Unix operating system. The default editor that comes with the UNIX operating system is called vi (visual editor). [Alternate editors for UNIX environments include pico and emacs, a product of GNU.] The UNIX vi editor is a full-screen editor and has two modes of operation: Command mode commands which cause the action to be taken on the file, and.

The VI editor is the most popular and classic text editor in the Linux family. Below, are some reasons which make it a widely used editor –

1) It is available in almost all Linux Distributions

2) It works the same across different platforms and Distributions

3) It is user-friendly. Hence, millions of Linux users love it

To open a file in the vi editor to start editing, simply type in 'vi <filename>' in the command prompt. To launch the VI Editor -Open the Terminal (CLI) and type vi <filename\_NEW> or <filename\_EXISTING> And if you specify an existing file, then the editor would open it for you to edit. Else, you can create a new file.

We can use the vi editor to edit an existing file or to create a new file from scratch. We can also use this editor to just read a text file.





**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 2**

**Repeat factor.**

**Answer -**

Many vi mode commands may be prefixed by a command repeat factor. This means that most character, change, delete, word, movement, and positioning commands may be preceded by a number which refers to the number of times the command should be repeated.

For example, j moves the cursor 1 character down, and 4j moves the cursor 4 characters down. In the commands listed below, examples of using the repeat factor are shown by placing the character n in front of a vi mode command. Not all vi mode commands are shown with a repeat factor; just a few, to remind you that it exists!

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 3**

**Input mode and insertion of text.**

**Answer -**

vi Editor Insert mode: This mode is for inserting text in the file. You can switch to the Insert mode from the command mode by pressing 'i' on the keyboard Once you are in Insert mode, any key would be taken as an input for the file on which you are currently working. To return to the command mode and save the changes you have made you need to press the Esc key

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 4**

**Saving text and quitting.**

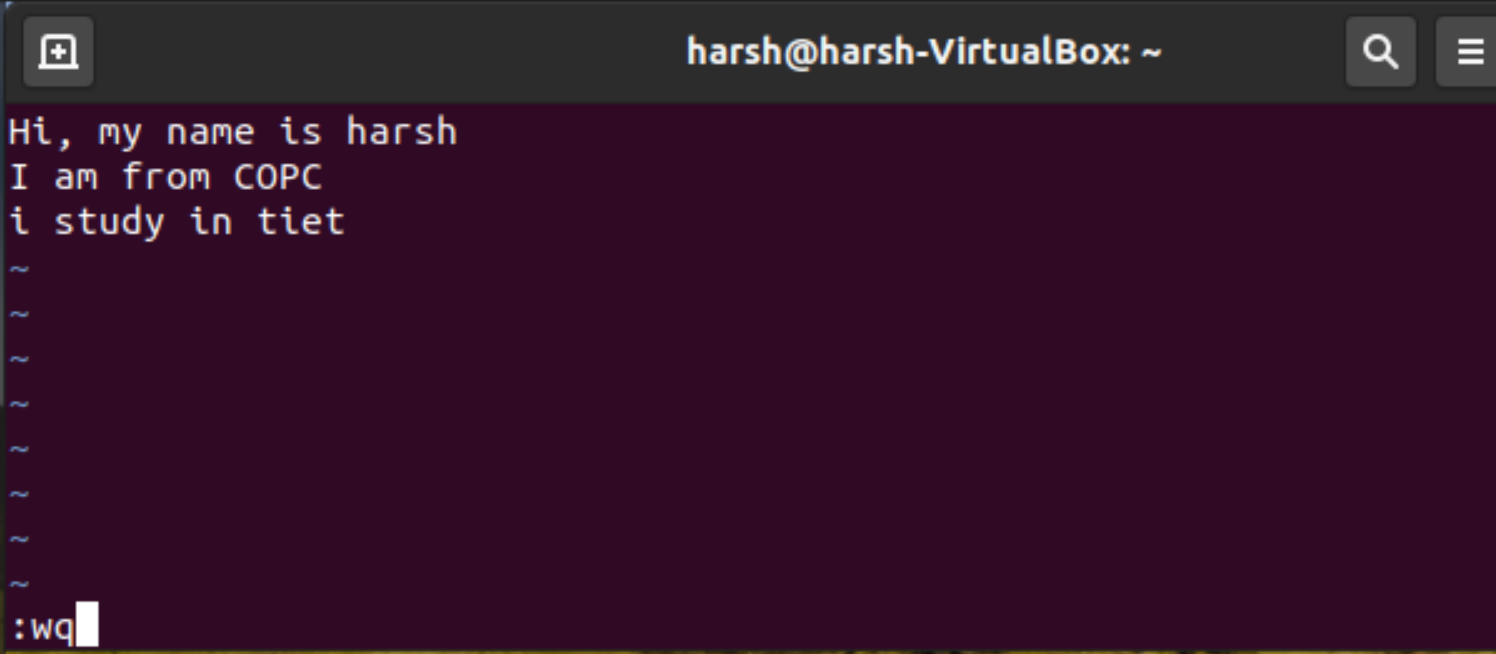
**Answer -**

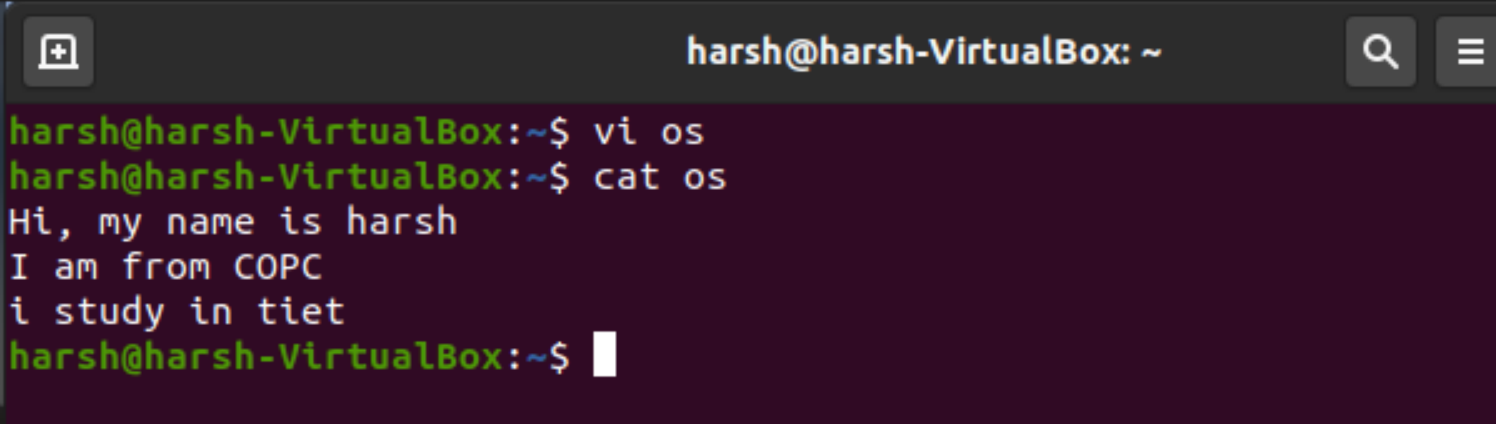
The procedure to save a file in vim / vi and quit the editor is as follows:

* Open the terminal application in Linux or Unix Next,
* open a file in vim / vi, type: vim filename
* To save a file in Vim / vi, press Esc key, type :w and hit Enter key
* One can save a file and quit vim / Vi by pressing Esc key, type :x and hit Enter key
* ownership may be granted to someone else by certain usersgroup - every file belongs to a single group. others - everyone else who is not in the group or is not the owner.
* Three permissions and three groups of people. That's about all there is to permissions really. Now let's see how we can view and change them.

**Saving and Closing the file**

* :w - Save the file but keep it open
* :q - Quit without saving
* :wq - Save the file and quit
* :x - saves and quits editing mode





**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 5**

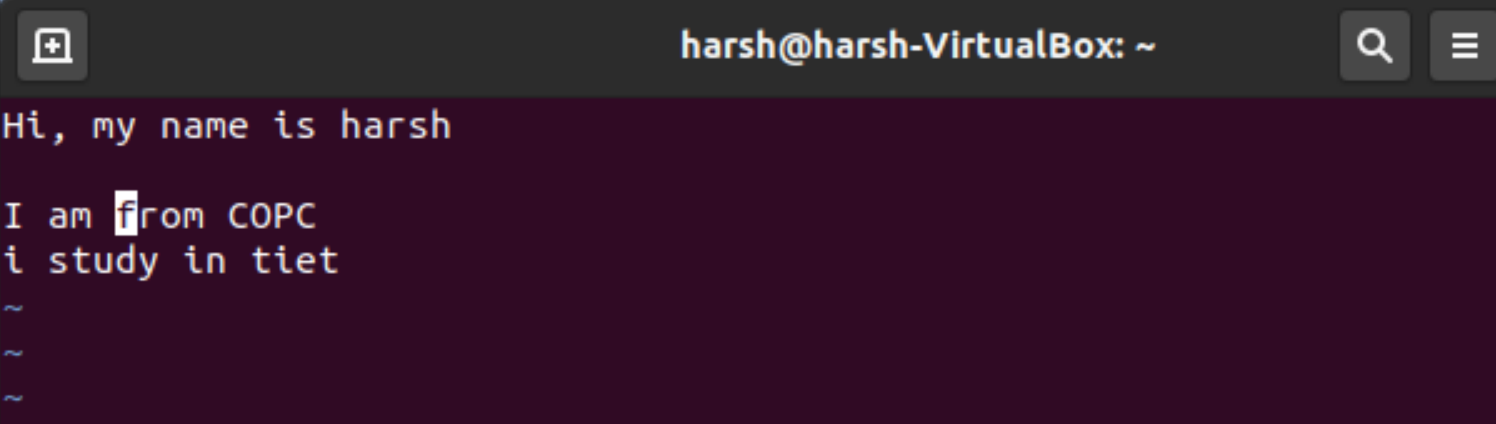
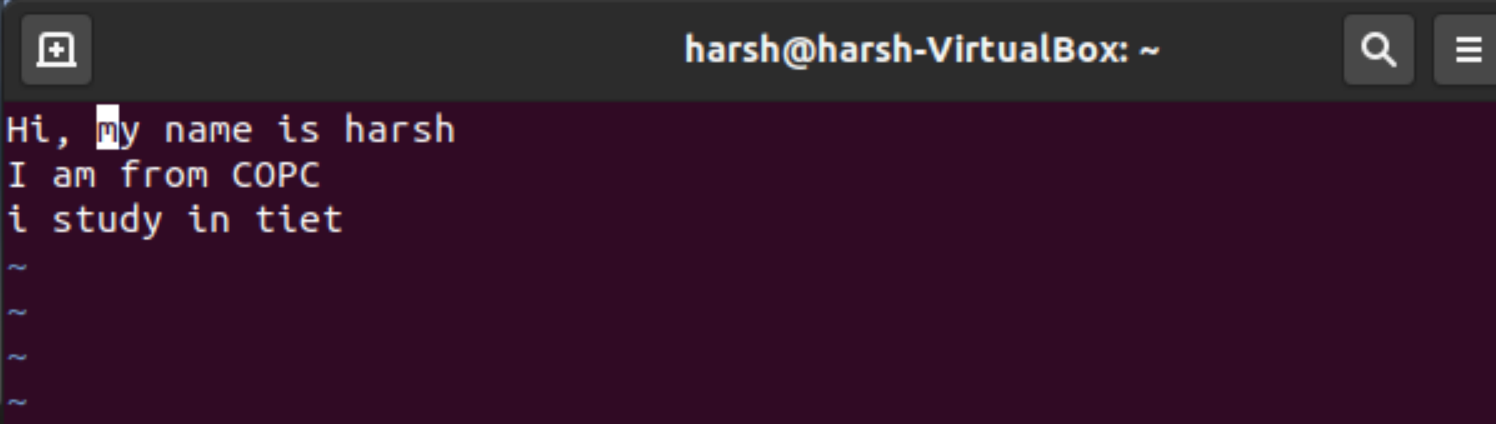
**Navigation**

**Answer -**

If you're using vi from a remote terminal, the arrow keys might not work correctly. The arrow key behaviour depends on your terminal emulator.

If the arrow keys don't work for you, you can use the following substitutes:

* To move left, press h.
* To move right, press l.
* To move down, press j.
* To move up, press k.
* Moving One Word Press w (“word”) to move the cursor to the right one word at a time.
* Press b (“back”) to move the cursor to the left one word at a time.
* Press W or B to move the cursor past the adjacent punctuation to the next or previous blank space.
* Press e (“end”) to move the cursor to the last character of the current word.
* Moving to Start or End of Line Press ^ to move the cursor to the start of the current line.
* Press $ to move the cursor to the end of the current line.
* Moving Down One Line Press the Return key to move the cursor to the beginning of the next line down.
* Moving Left - Press the BackSpace key to move the cursor one character to the left.
* Moving Right - Press the Space Bar to move the cursor one character to the right.
* Moving to the Top - Press H (“high”) to move the cursor to the top of the screen.
* Moving to the Middle - Press M (“middle”) to move the cursor to the middle of the screen.
* Moving to the Bottom - Press L (“low”) to move the cursor to the bottom of the screen.



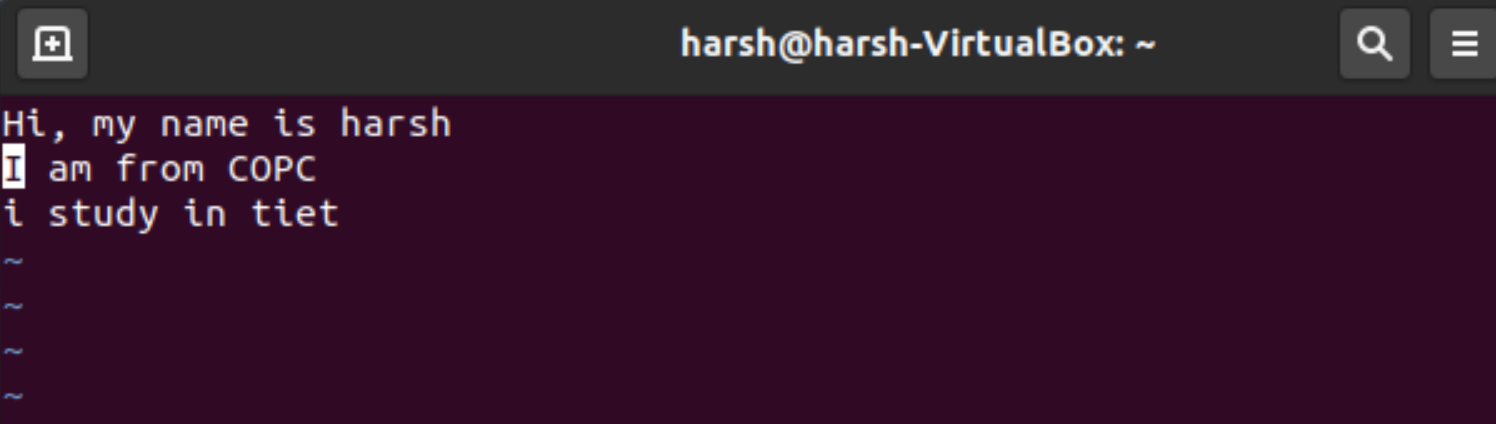
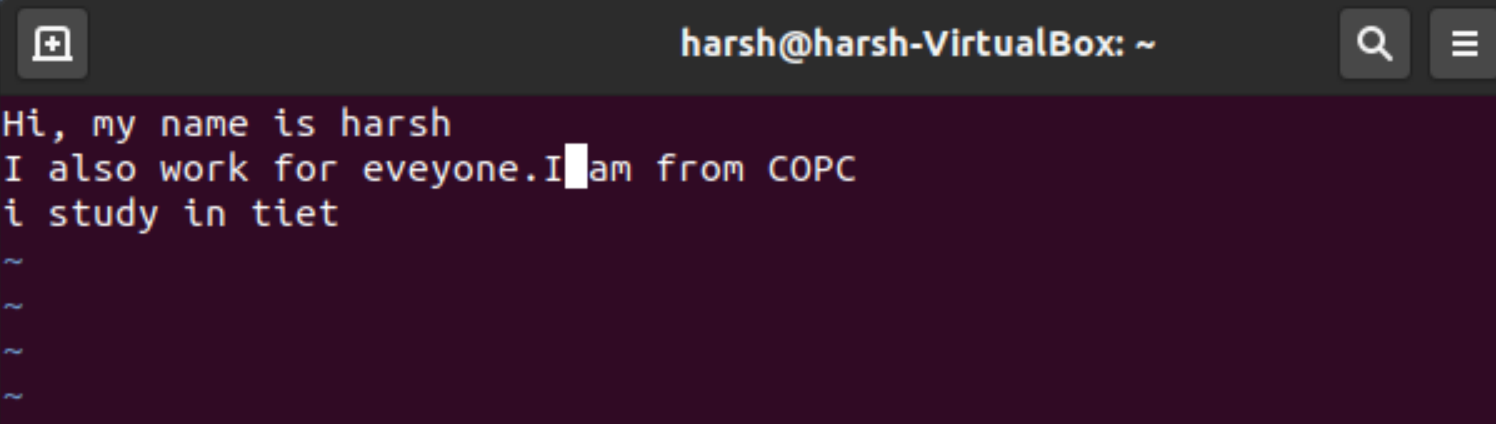
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

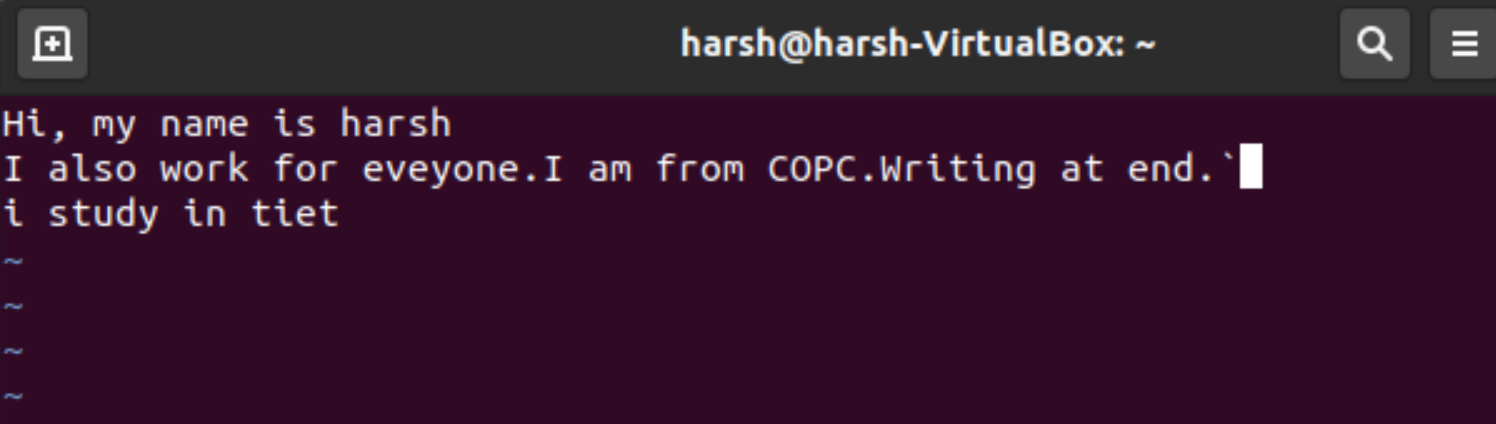
**Question 6**

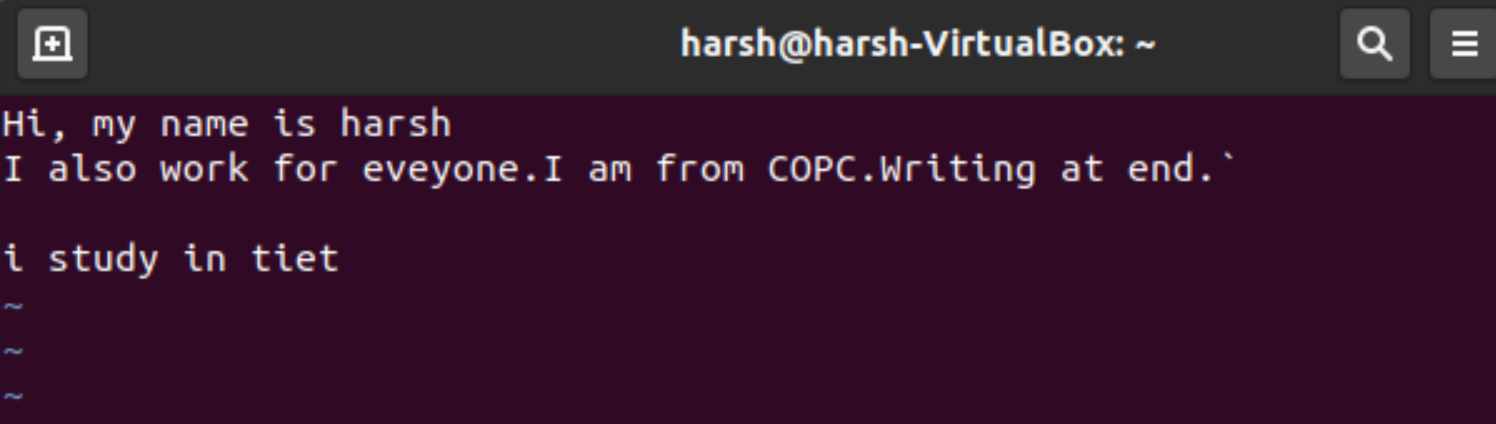
**Editing text**

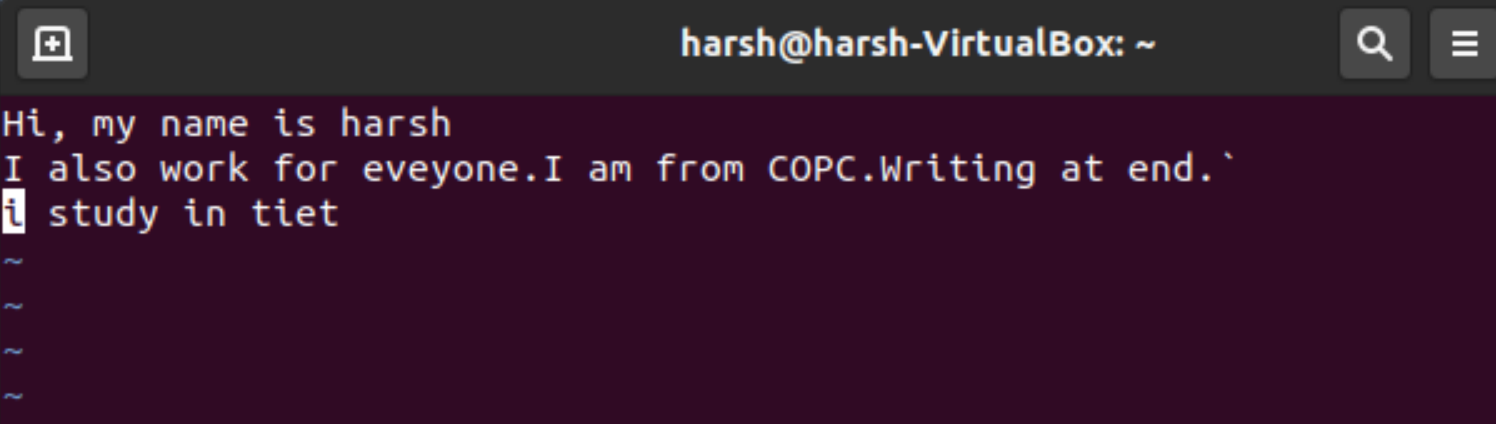
**Answer -**

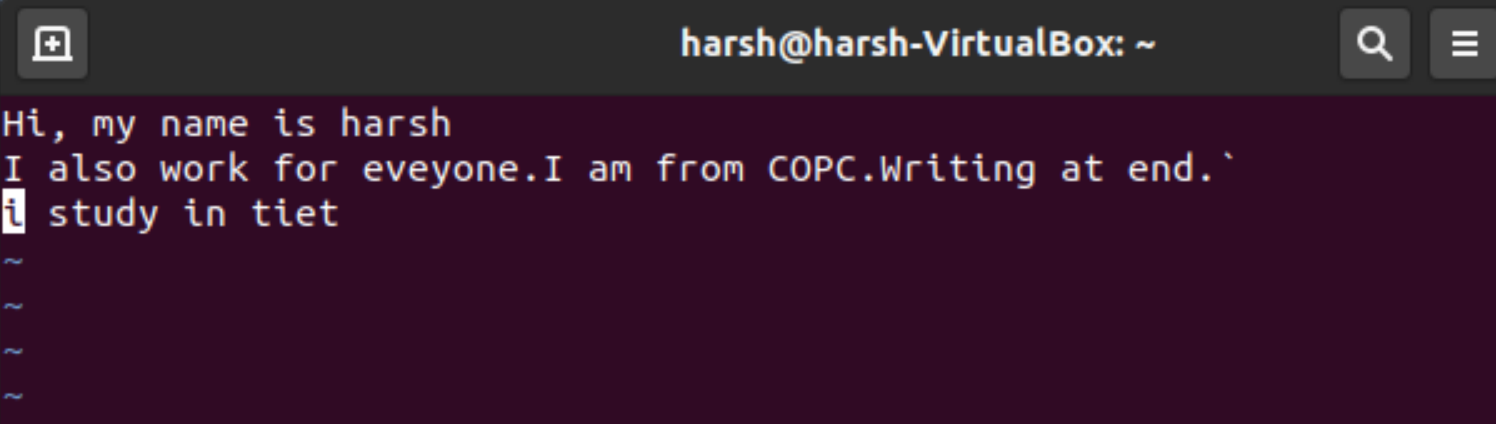
**VI Editing commands**

* i - Insert at cursor (goes into insert mode) 
* a - Write after cursor (goes into insert mode) 
* A - Write at the end of line (goes into insert mode)

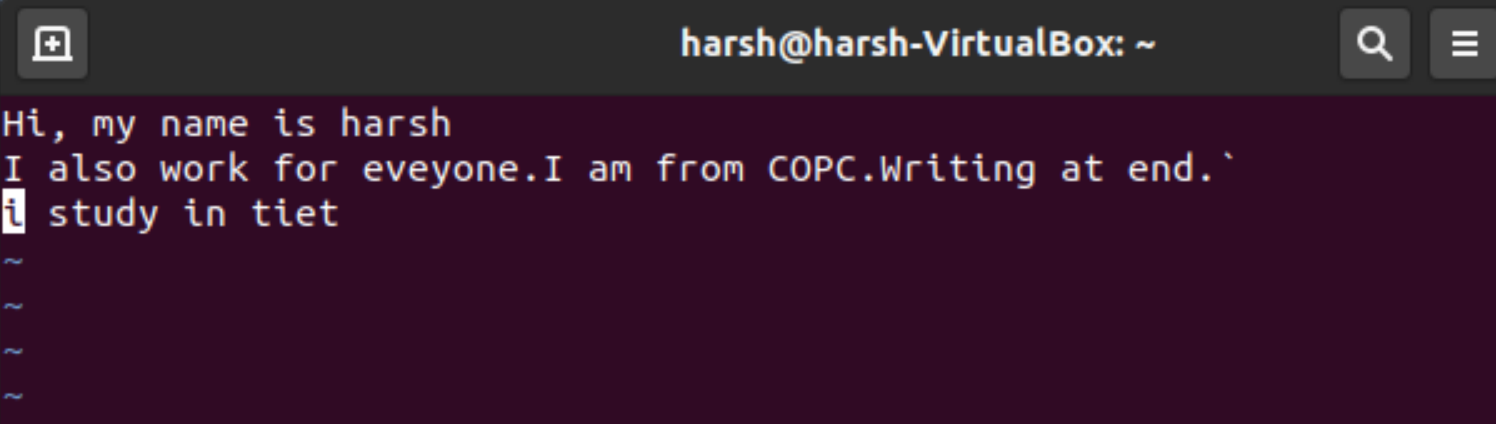


* ESC - Terminate insert mode
* o - Open a new line (goes into insert mode) 
* dd - Delete single line

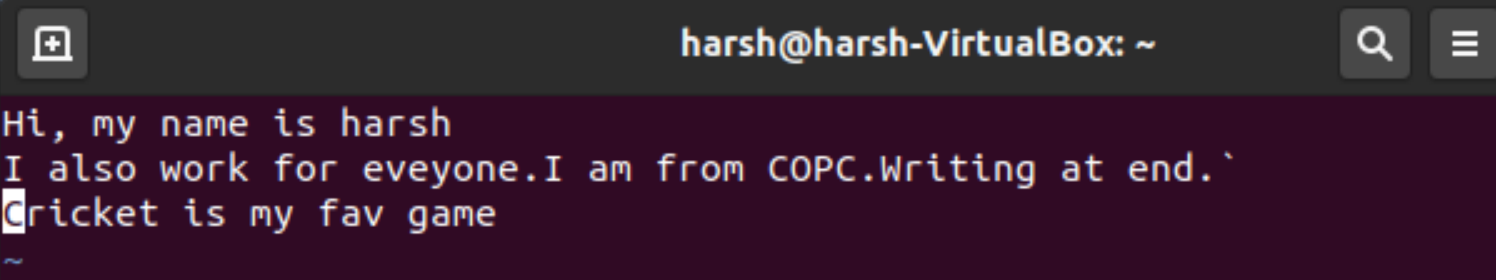


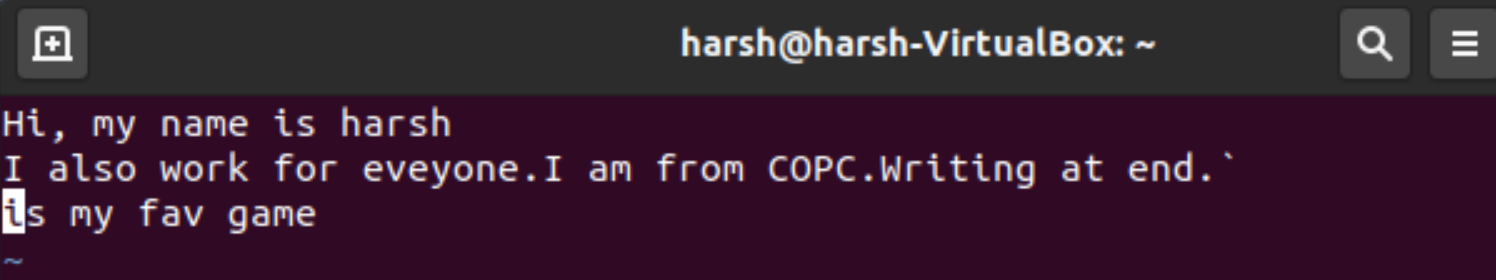


* D - Delete contents of the line after the cursor

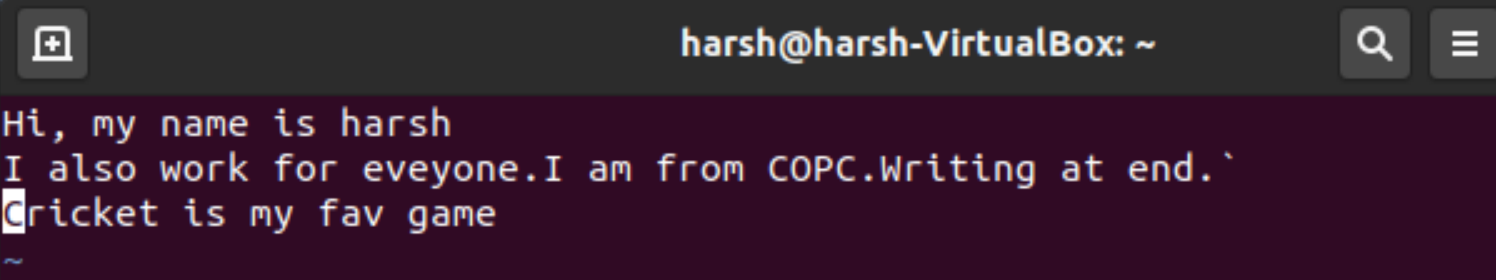


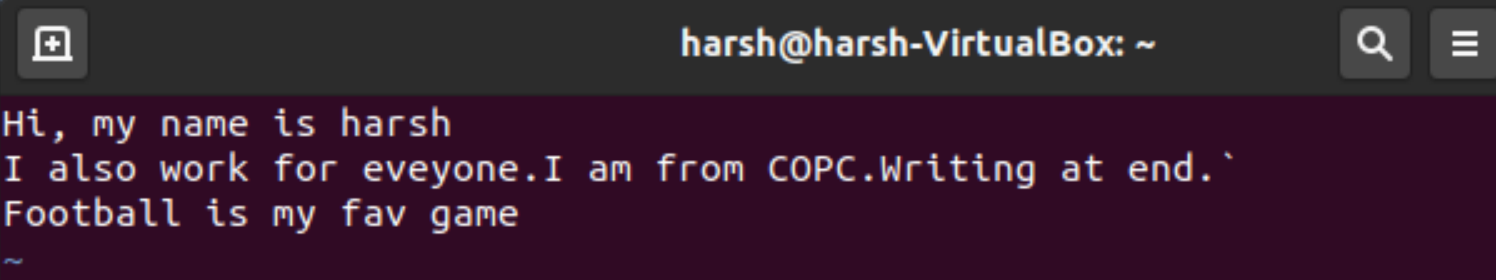


* dw - Delete word 

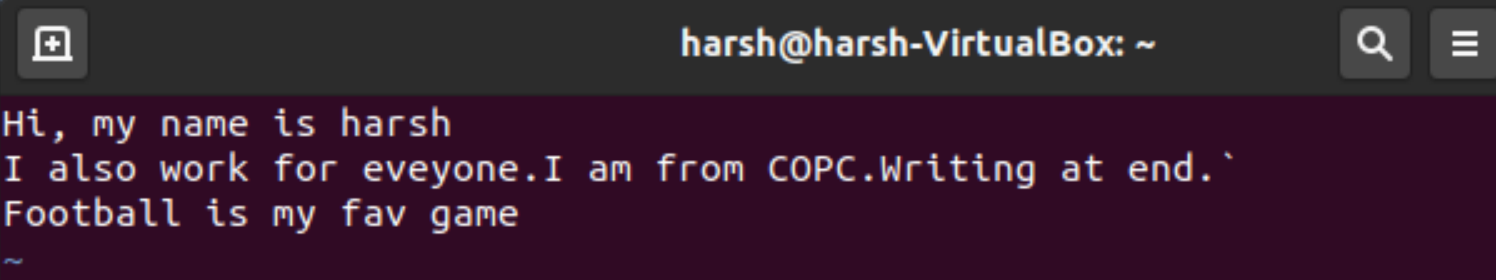


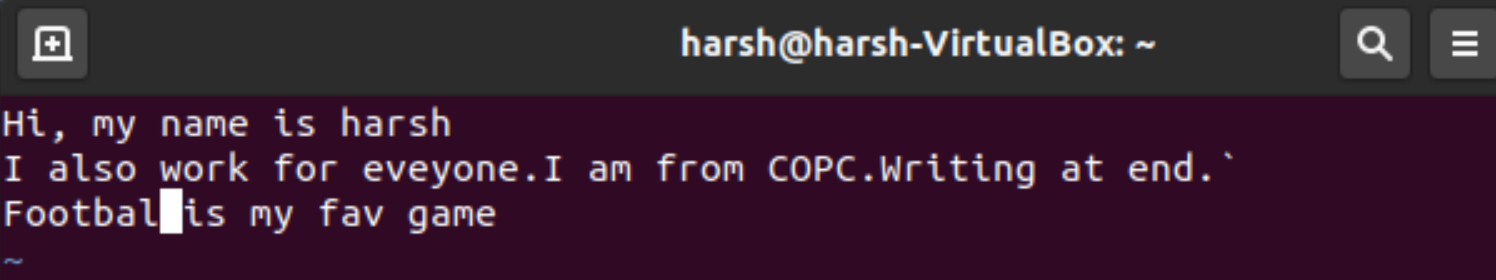
* cw - Change word





* x - Delete character at the cursor





**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 7**

**Undo the last editing.**

**Answer -**

u - Undo last change

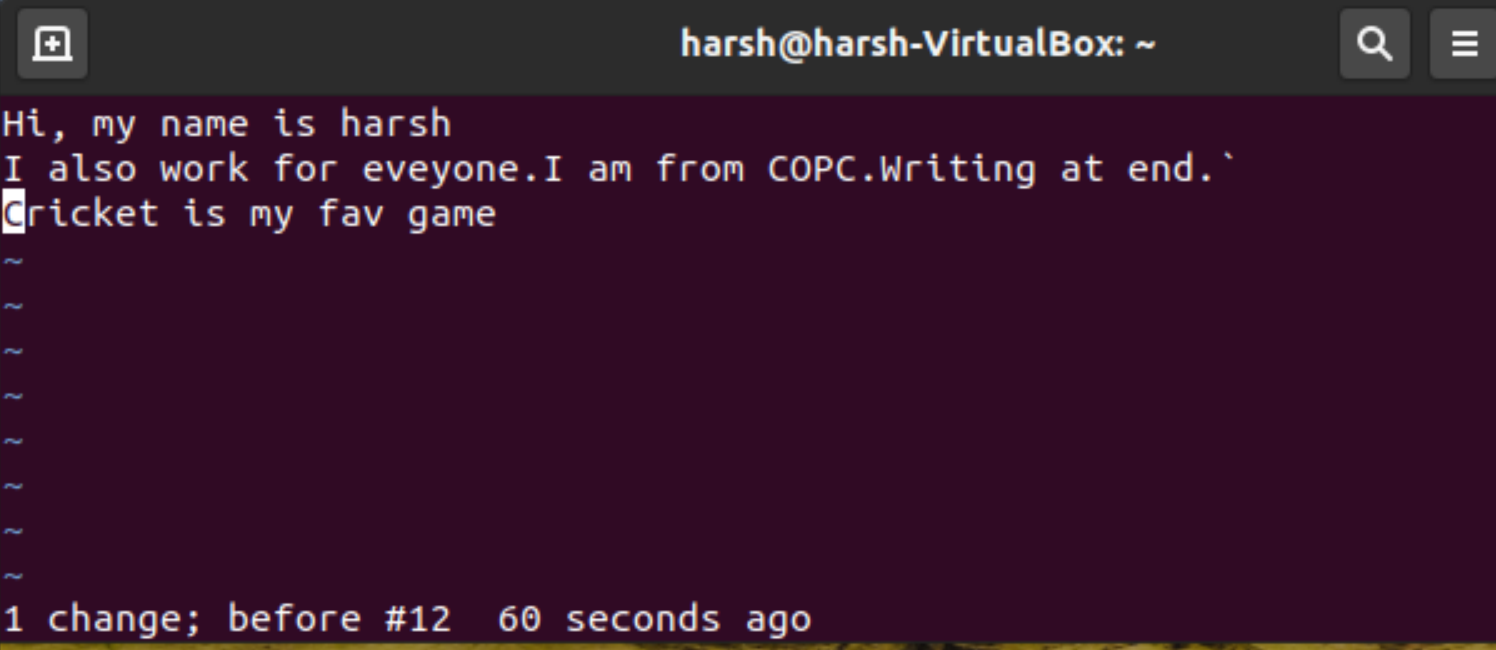
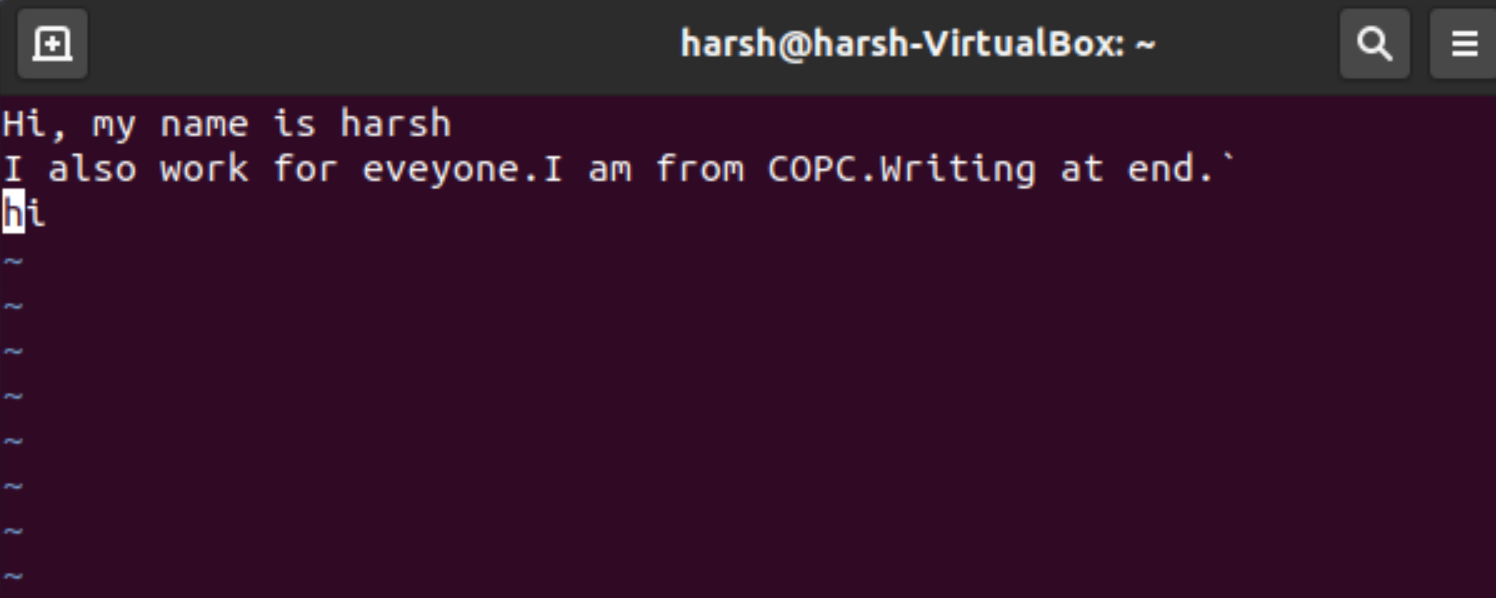
U - Undo all changes to the entire line

In command mode,

* To undo the last change made, we use : ‘u’
* To discard all changes made to the current line, we use: ‘U’

vim (LINUX) lets us undo and redo multiple editing instructions.

* ‘u’ behaves differently here; repeated use of this key progressively undoes our previous actions.
* 10u reverses our last 10 editing actions. The function of U remains the same.



**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

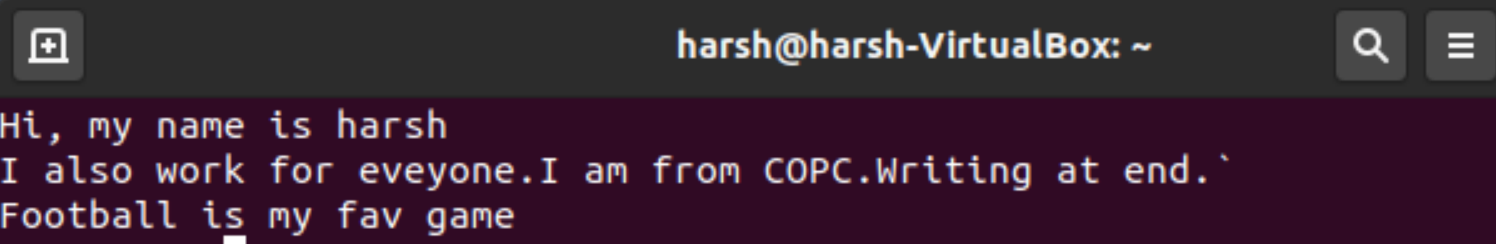
**Question 8**

**Repeating last command.**

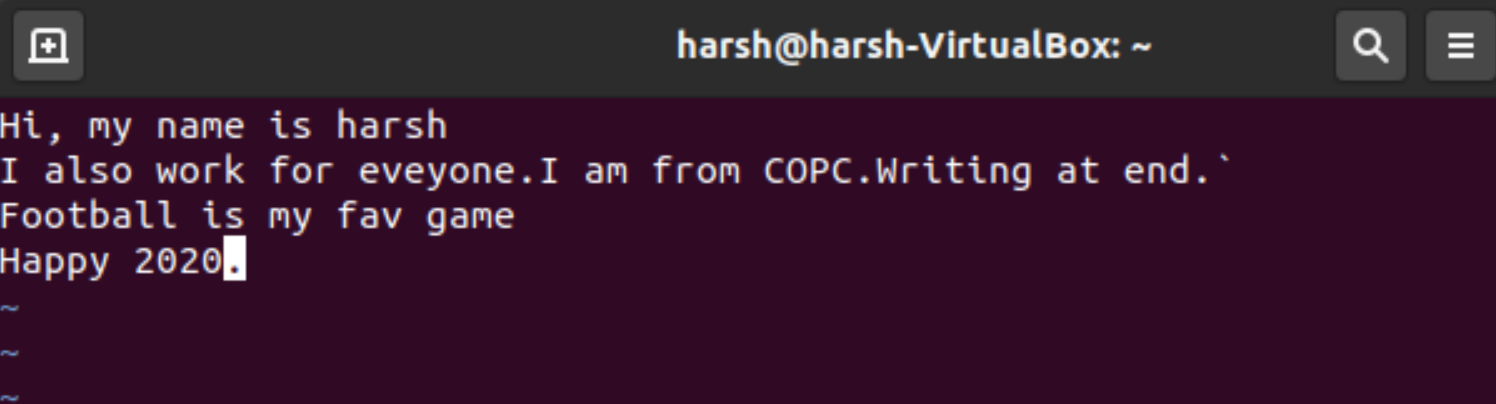
**Answer -**

The ‘.’ (dot) command is used for repeating the last instruction in both editing and command mode commands. For example 2dd deletes 2 lines from current line and to repeat this operation, we type ‘.’ (dot)

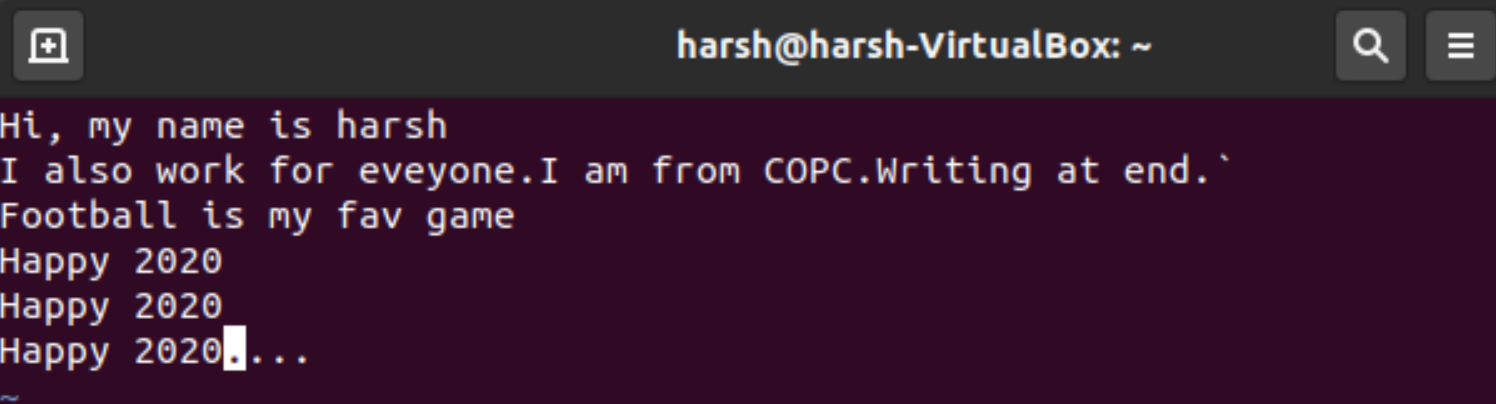
Before, doing any operation.



After the last operation. The last operation did is “Happy 2020”



Now, every time I use dot(.) command it keeps on adding “Happy 2020”.



**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 9**

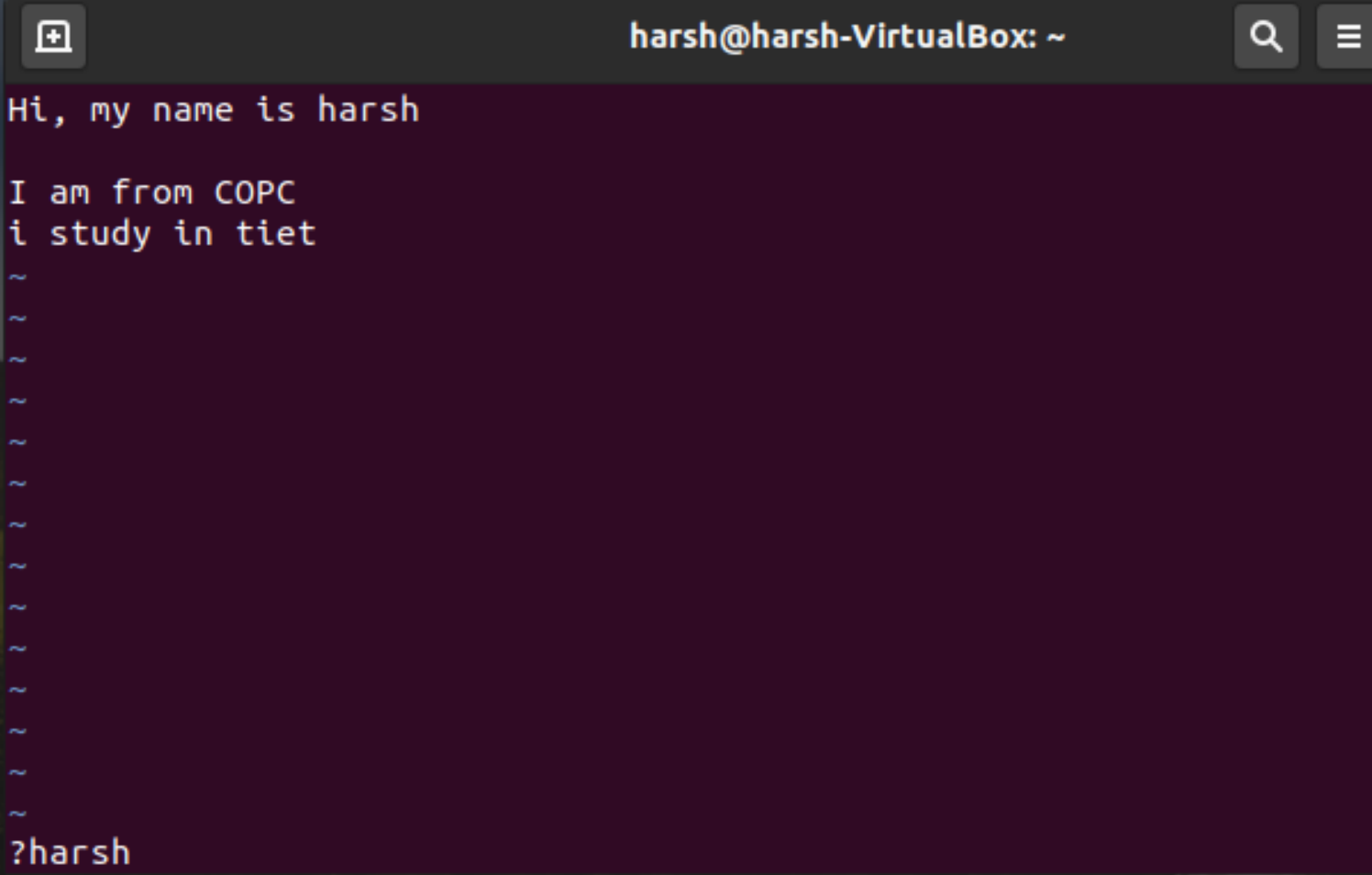
**Searching for a pattern.**

**Answer -**

To search using Vim/vi, for the current word:

* In normal mode, you can search forward or backward.
* One can search forward in vim/vi by pressing / and then typing your search pattern/word.
* To search backward in vi/vim by pressing ? and then typing your search pattern/word.

These are some of the commands used for searching for a pattern.

* / : search forward. ?
* 
* ? : search backwards. 

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 10**

**Substitution – search and replace**

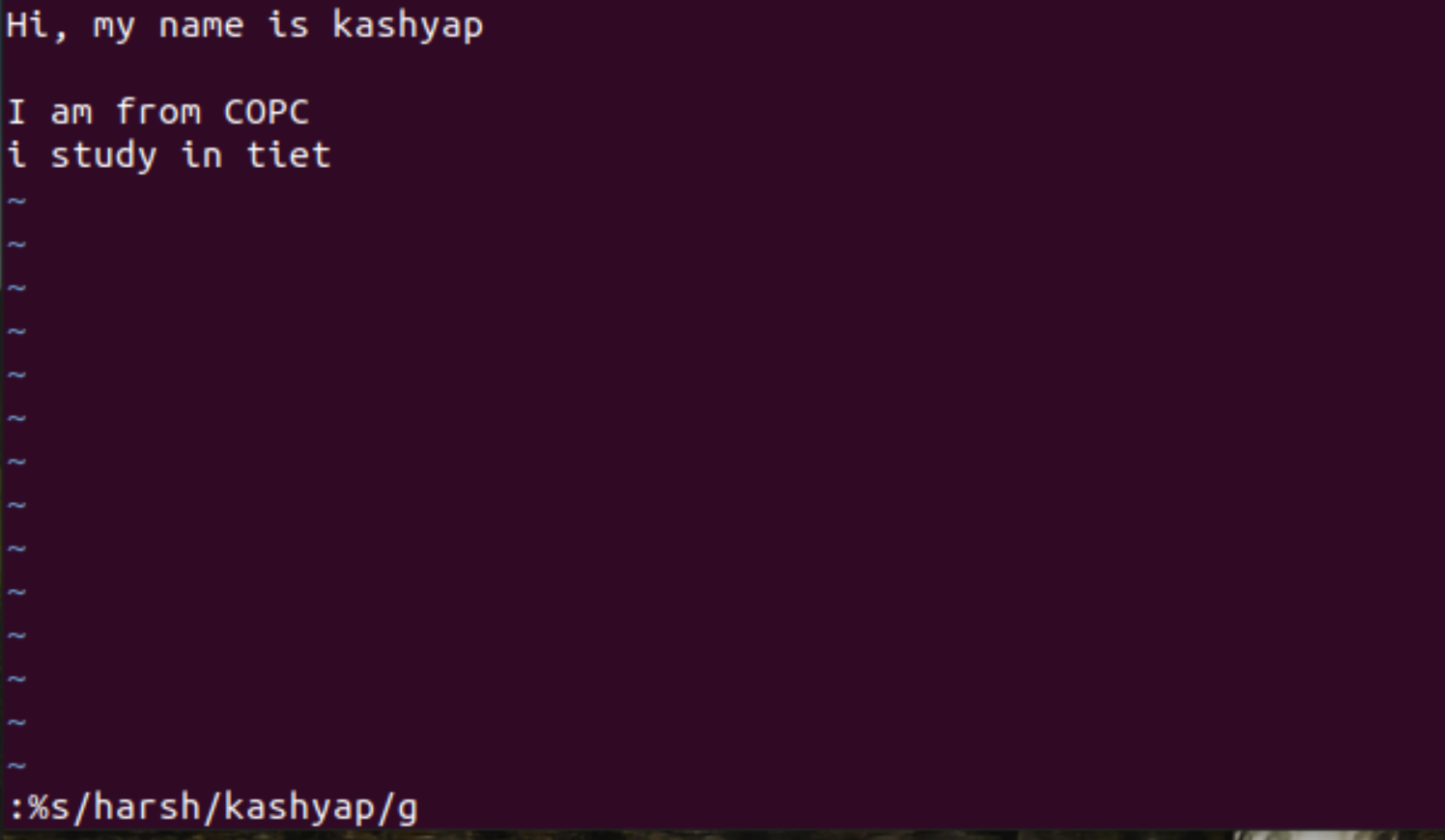
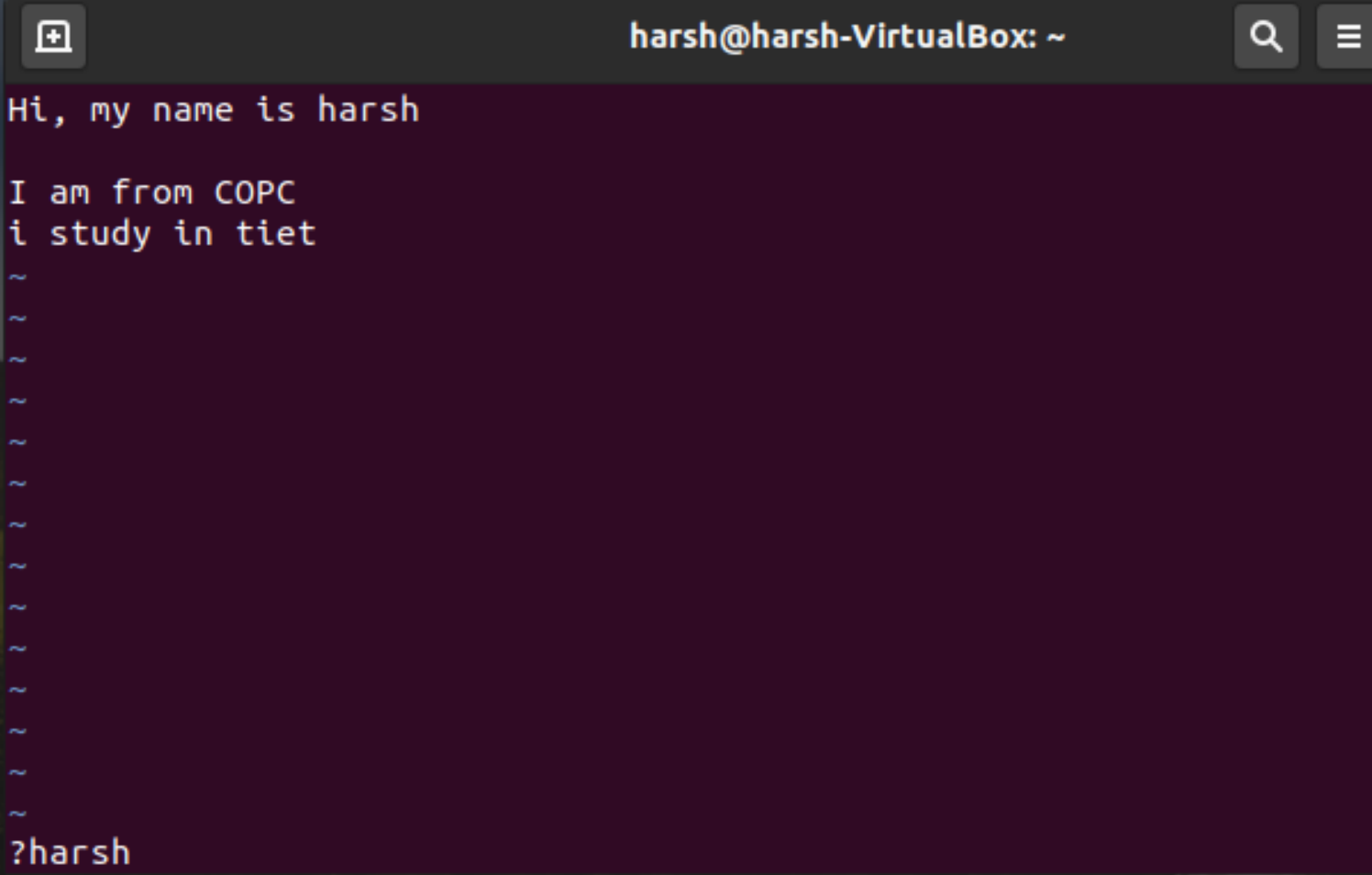
**Answer -**

The :substitute command searches for a text pattern, and replaces it with a text string.

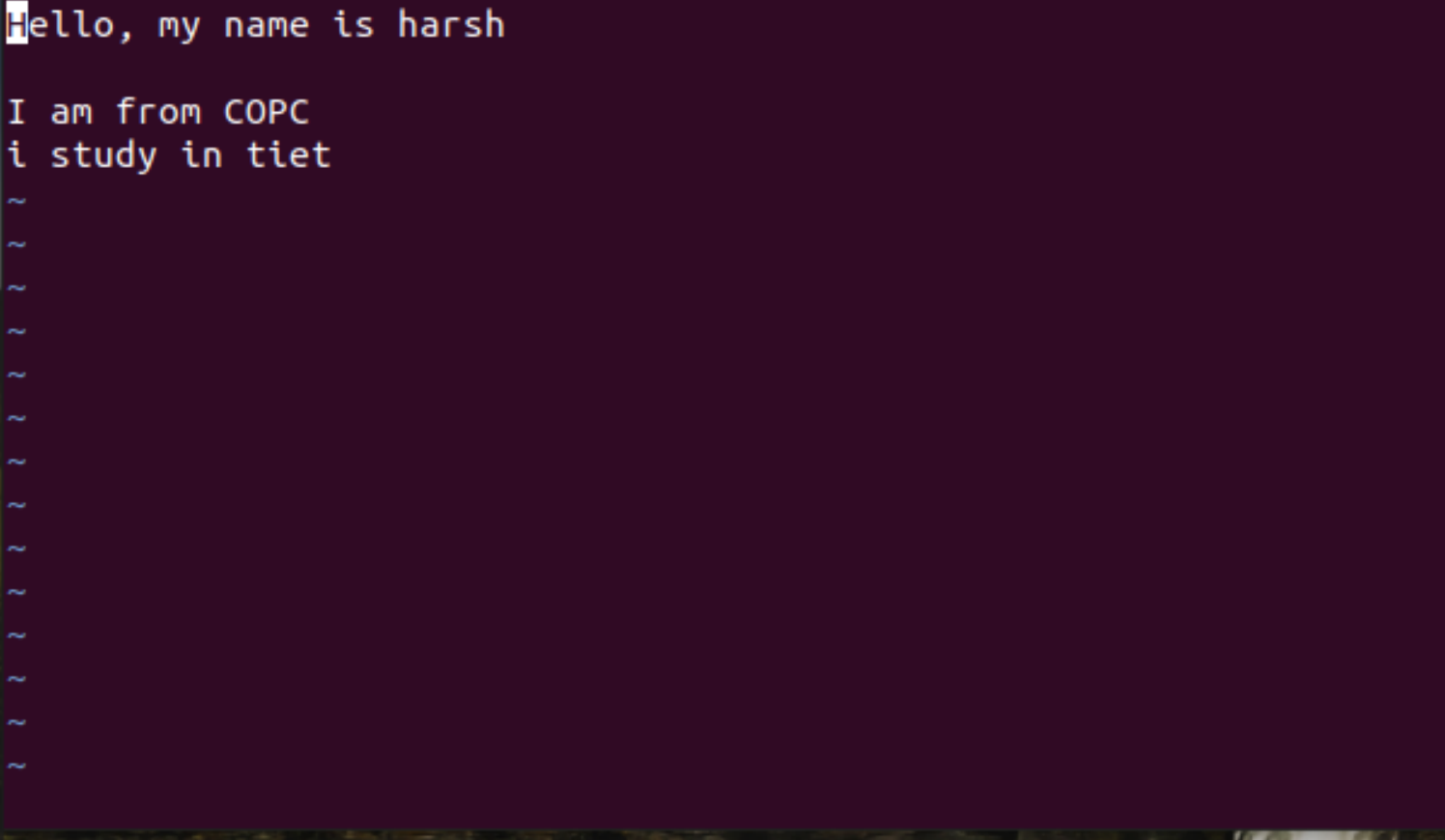
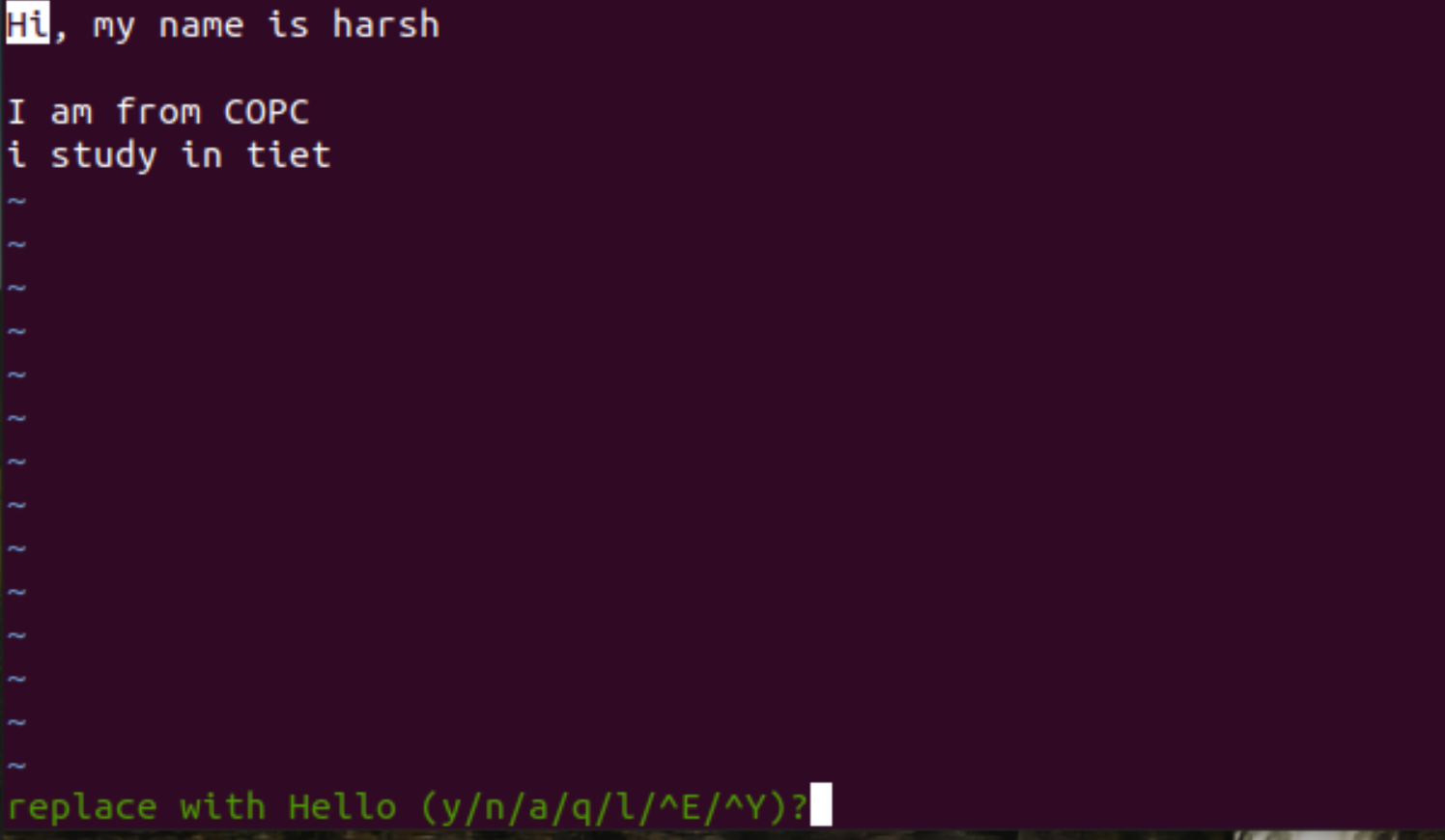
There are many options, but these are what you probably want:

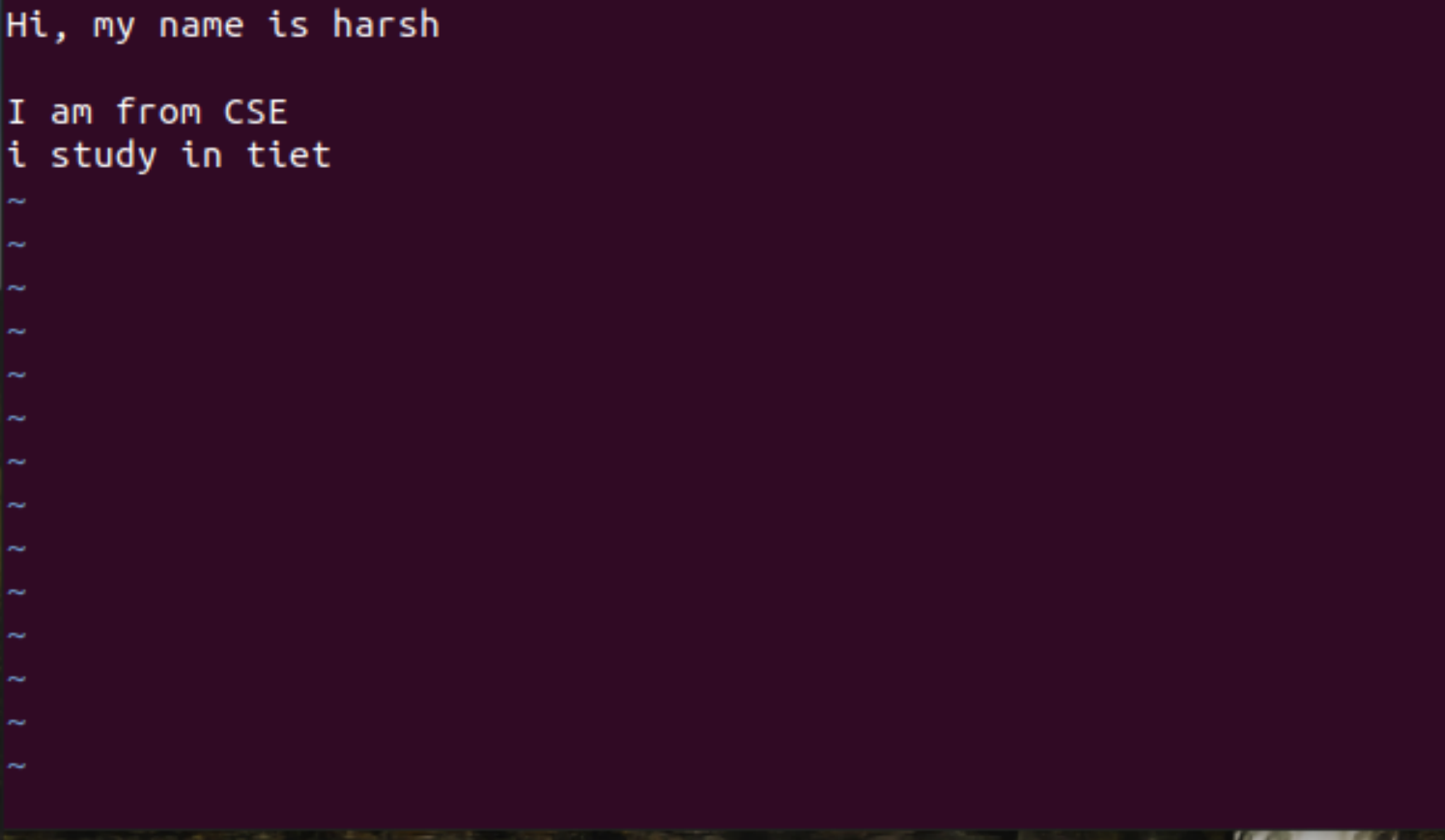
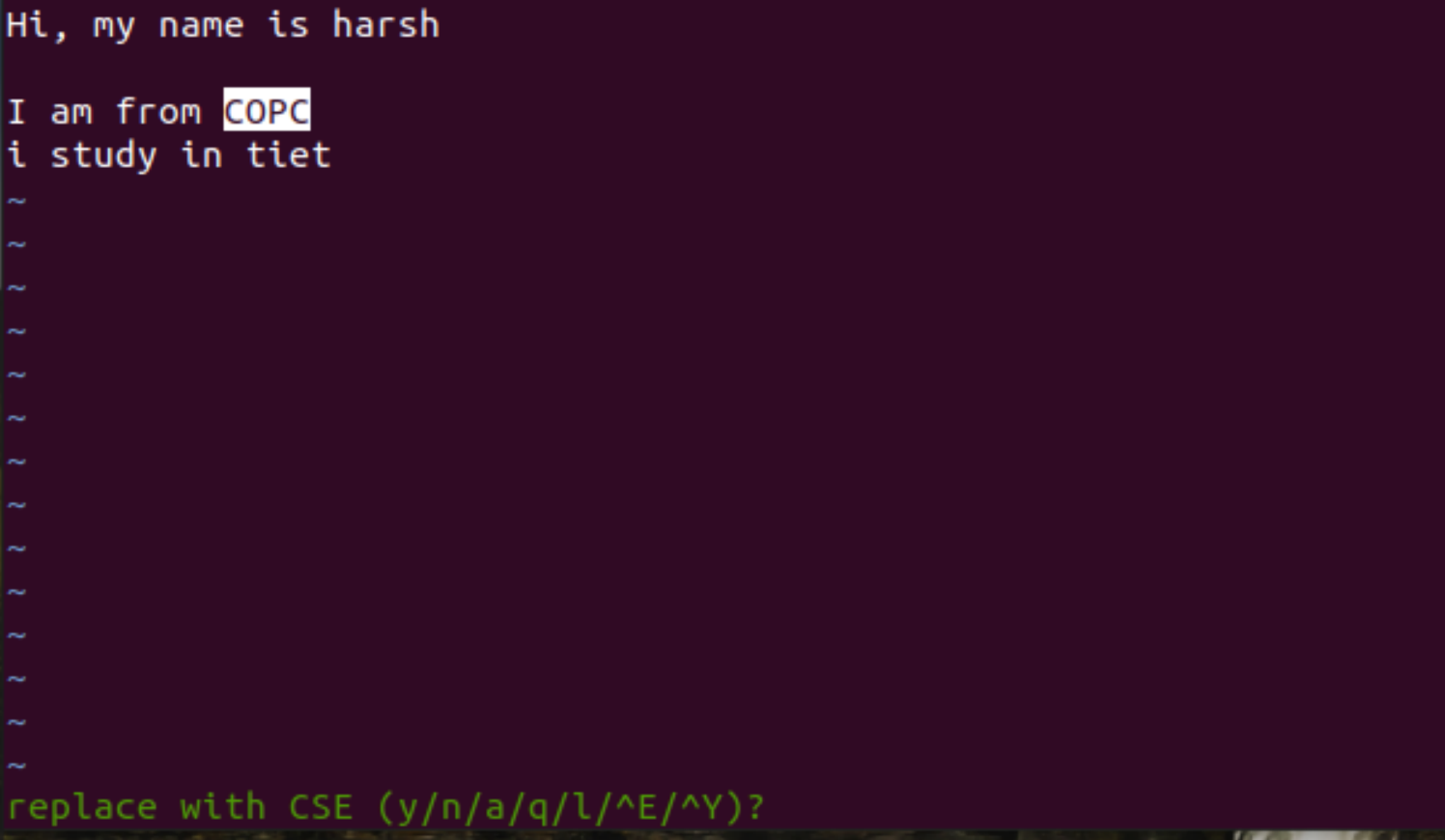
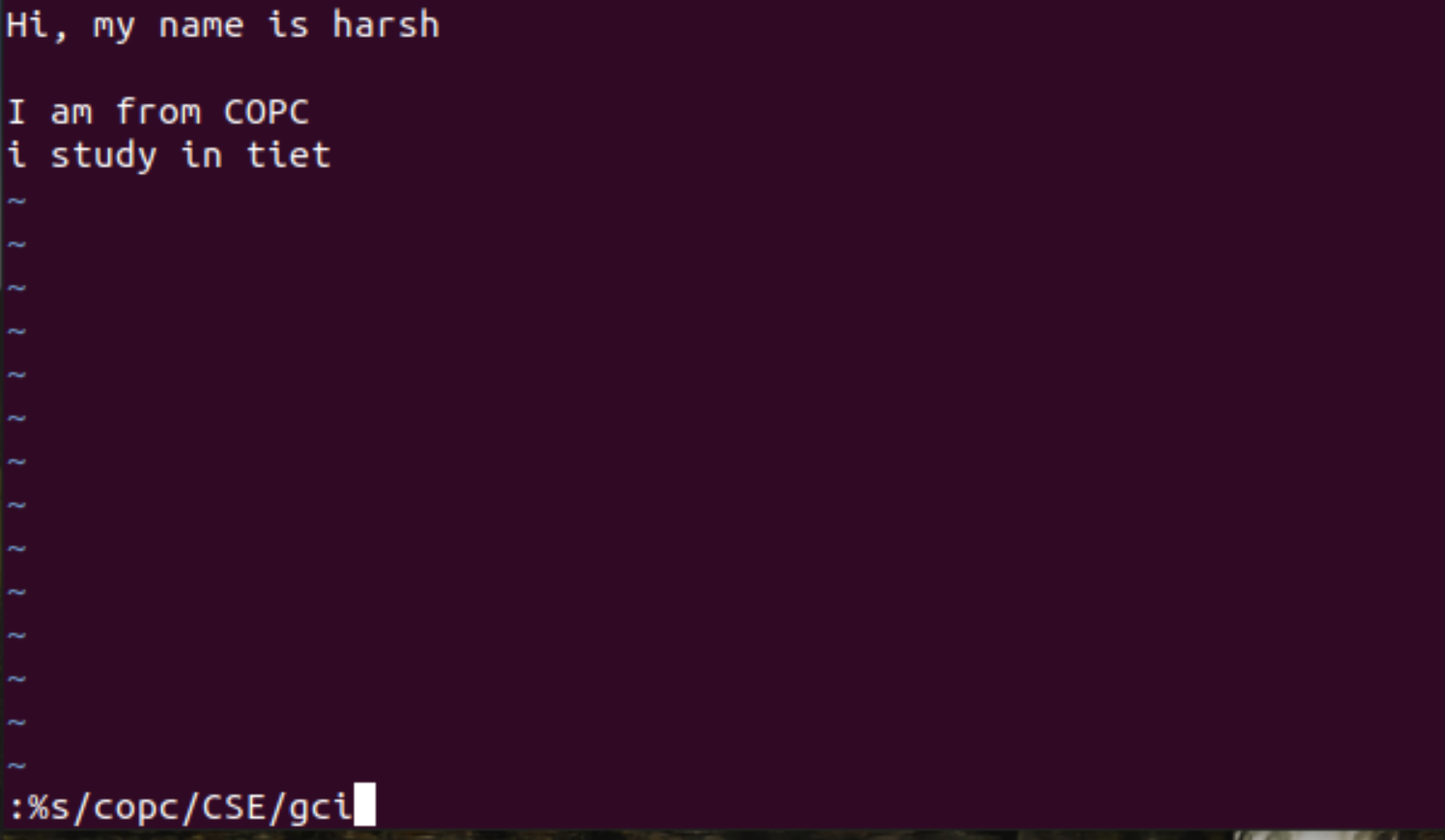
* :%s/pattern\_old/new\_pattern/g - Find each occurrence of 'pattern\_old' (in all lines), and replace it with 'new\_pattern'.

For example, here I have replaced harsh with kashyap.



* :%s/old\_pattern/new\_pattern/gc - Change each “old\_pattern” to 'new\_pattern', but ask for confirmation first.



* :%s/old\_pattern/new\_pattern/gci - Change each 'old\_pattern' (case insensitive due to the i flag) to 'new\_pattern' but asks for confirmation. 

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**